NMS and Thresholding Architecture used for FPGA based Canny Edge Detector for Area Optimization

[1,]Chandrashekar N.S.,[2,] Dr. K.R. Nataraj

[1,]Department of ECE, Don Bosco Institute of Technology, Bangalore,Karnataka, India.

chandrashekarns@rediffmail.com

[2,]Department of ECE, SJB Institute of Technology, Bangalore, Karnataka, India.

Nataraj.sjbit@gmail.com

***Abstract*—— In this paper, an architecture designed for Non-Maximal Suppression used in Canny edge detection algorithm is presented in order to reduce memory requirements significantly. The architecture also achieves decreased latency and increased throughput with no loss in edge detection. The new algorithm used has a low-complexity 8-bin non-uniform gradient magnitude histogram to compute block-based hysteresis thresholds that are used by the Canny edge detector. Furthermore, the hardware architecture of the proposed algorithm is presented in this paper and the architecture is synthesized on the Xilinx Virtex 5 FPGA. The design development is done in VHDL and simulated results are obtained using modelsim 6.3 with Xilinx 12.2.**

***Keywords* — Canny Edge Detector, Distributed Processing, Non-uniform Quantization, FPGA .**

I. Introduction

The analysis of image processing is simplified by the edge detection process, by means of reducing the amount of data to be processed, while preserving useful structural information about object at the same time. There is certainly a great deal of diversity in the applications of edge detection, but it is felt that many applications share a common set of requirements [1]. The Canny edge detector is used in many real-world applications due to its ability to extract significant edges with good detection and good localization performance. Unfortunately, the canny edge detection algorithm contains extensive pre-processing and post-processing steps and is more computationally complex than other edge detection algorithms, such as Roberts, Prewitt and Sobel algorithms [2].

In [1, 8], a recursively implementable edge detection algorithm is suggested and optimized, using retiming techniques. But its performance is quite poor in images with low SNRs. The approach of [5, 6] combines the derivative and smoothening operations of the Canny algorithm into a single mask to reduce computations. The pipelined implementation is a block-based approach with a block-size of 2 rows of pixels. It overcomes the dependencies between the blocks by fixing high and low thresholds to a constant value. In both of these approaches gradient thresholds are not adapted to the image characteristics, and their performance is not guaranteed for blurred images and images with low SNRs [3]. In [4] Canny edge detector a parallel architecture of simultaneous 4-pixel calculation is proposed, which increases the throughput of the design without increasing the need for on-chip cache memories. This design has been synthesized for low-end and high-end Xilinx FPGA. However, in [3] the hysteresis thresholds calculation is based on a very finely and uniformly quantized 64-bin gradient magnitude histogram, which is computationally expensive and, thereby, hinders the real-time implementation. In this paper, a method based on non-uniform and coarse quantization of the gradient magnitude histogram is proposed. In addition, the proposed algorithm is mapped onto reconfigurable hardware architecture. This architecture exploits the parallelism and pipelining of the proposed algorithm, and therefore, yields significant speedup in running times. In [7], GPU was used due to its powerful ability of parallel processing while a new Canny operator is proposed with the introduction of parallel breakpoints detection and edge tracing without recursive operations.

In [9], an improvised approach is done for image edge detection and its efficiency with a set of images. Results confirm that the DOW-canny edge detector, besides being insensitive to noise, is able to fabricate superior and accurate edges in regions of fine graining, geometrical figures and alpha-numerics as compared to the current edge detection techniques. The detector is also able to handle noisy as well as noiseless images.

This paper is organized as follows: Section 2 gives a Brief analysis of the Canny edge detector algorithm. Section 3 presents the proposed Canny edge detection algorithm which includes a novel method for the hysteresis thresholds computation based on a non-uniform quantized gradient magnitude histogram. Simulation results are presented in Section 4. A conclusion is given in Section 5.

II. Canny Algorithm Analysis

The block diagram of the Canny algorithm is demonstrated in Fig. 1. The Canny algorithm first smoothens the image to eliminate noise by using a smoothing filter such as the Gaussian Convolution. The Gaussian smoothing was performed using a mask (matrix) which was sled over the image, manipulating a square of pixels at a time. The bigger the dimensions of the mask, lower was the sensitivity the detector had to noise. A mask was a common choice for the size of a Gaussian filter. The output smoothed image was denoted as I (x, y).

After smoothing the next step was the calculation of gradient of the image. The gradient calculation lead to the

DOI: 03.LSCS.2013.2.26

detection of the possible edge strength and direction. This was executed by another convolution with a gradient operator. The most commonly used gradient operators are the Prewitt and the Sobel operators. Both operators perform a 2-D spatial gradient measurement on an image. Calculating the horizontal gradient Gx(x, y) and vertical gradient Gy(x, y) at each pixel location by convolving the image I(x, y) with partial derivatives of a 2D Gaussian function. Computing the gradient magnitude G(x, y) and direction θ(x, y) at each pixel location. Gradient magnitude and orientation are given by the following equations:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}} \quad (1)$$

$$\theta = \arctan (G_y/G_x) \quad (2)$$

Arctan and the division can be eliminated by simply comparing Gx and Gy values, if they are of similar length, a diagonal direction is obtained, if one is at least 2.5 times longer than the other, a horizontal or vertical direction is obtained.

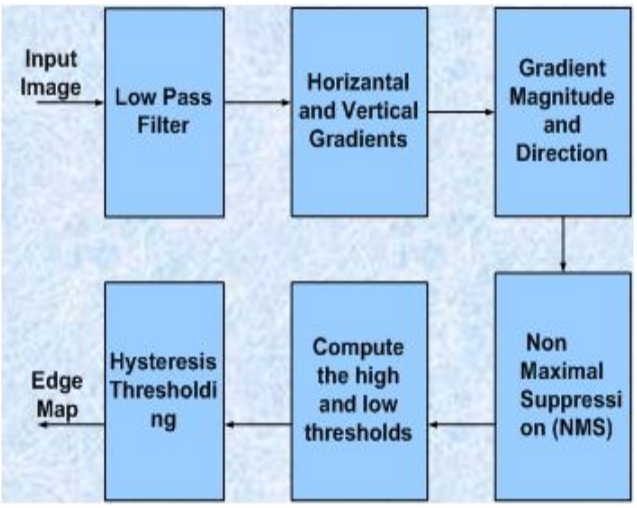

Figure 1. Block diagram of the Canny edge detection algorithm

After the edge directions were known, non-maximum suppression was applied. Non-maximum suppression is used to trace pixels along the gradient in the edge direction and compare their values perpendicular to the gradient. Two perpendicular pixel values were compared with the value in the edge direction. If their value was lower than the pixel on the edge then they were suppressed i.e., their pixel value was changed to 0, else the higher pixel value was set as the edge and the other two suppressed with a pixel value of 0. Finally, hysteresis was used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and was marked as such immediately. Then, any pixels that were connected to this edge pixel and having a value greater than T2 were also selected as edge pixels. If an edge was to be followed, a gradient of T2 is needed to start but it doesn't stop till a gradient below T1 [1] is hit.

III. Implementation Of The Proposed Canny Edge Detector

In this section, the hardware implementation of proposed canny edge detection algorithm on the Xilinx FPGA is described. The proposed architecture consists of the following 5 units:

- Smoothening unit using Gaussian filter.
- Vertical and horizontal gradient Magnitude calculation unit.
- Directional non-maximum suppression unit.
- High and low threshold Calculation unit.
- Thresholding with hysteresis unit.

A. Image Smoothening:

The input image was smoothened using a 3×3 Gaussian mask. The Gaussian filter is separable and, thus, the implementation of the 2-D convolution with the 3×3 Gaussian mask was achieved using row and column 1- D convolutions.

B. Gradients and Gradient Magnitude Calculation:

Calculation of the vertical and horizontal gradients using convolution kernels was done in this stage. The kernels vary in size from 3 × 3 to 9 × 9, depending on the sharpness of the image. The whole design was pipelined, and thus the output was generated at every clock cycle. This was an input to the magnitude calculation unit which computes, at each pixel location, the gradient magnitude from the pixel's horizontal and vertical gradients.

C. Directional Non Maximum Suppression:

The non-maxima elimination filter is used to eliminate pixels that are not part of a continuous line. In other the pixels which are having high gradient magnitude but they are not a part of a continuous line can be easily eliminated using this filter. General concept used for the elimination of the unwanted pixel is as follows. A simple 3x3 window for the given image was considered and the value of the center pixel in comparison with the adjacent pixel in the given window that are perpendicular to the center pixel was checked. When the comparison was found to be too high, that is if the center pixel value was too high as compared to the adjacent pixel, the center pixel was eliminated and replaced with the earlier pixel value. Fig. 2 given below shows the basic 3x3 window and the pixel considered for the comparison. In Fig. 3, the architecture of the directional non-maximum suppression is shown. In order to access all the pixels' gradient magnitudes in the 3× 3 window at the same time, two FIFO buffers were employed. The horizontal gradient Gx and the vertical gradient Gy controlled the selector which delivers the gradient magnitude (marked as M(x, y) in Fig. 3) of neighbors along the direction of the gradient, into the arithmetic unit. This arithmetic unit consisted of one divider, two multipliers and one adder, which were implemented using the Xilinx Math

DOI: 03.LSCS.2013.2.26

Functions IP cores. The output of the arithmetic unit was compared with the gradient magnitude of the center pixel, and the pixel that had no local maximum gradient magnitude was eliminated.

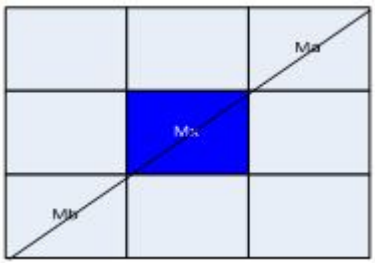

Figure 2. 3x3 window for non-maximum suppression unit

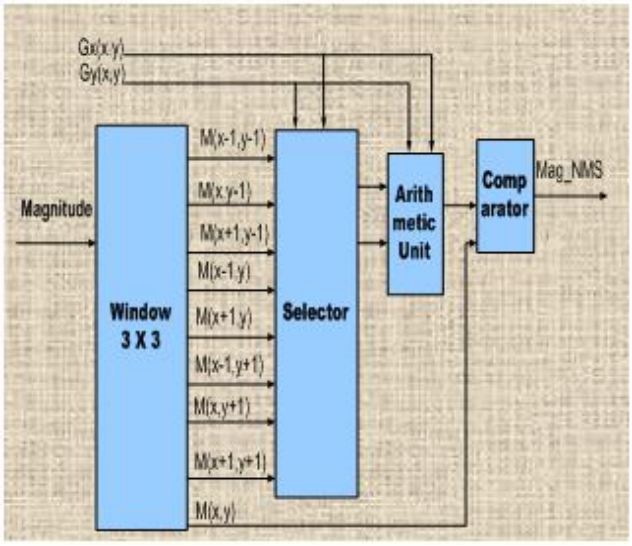

Figure 3. Directional Non Maximum Suppression Unit Architecture

D. Calculation of the hysteresis thresholds:

Since the low and high thresholds were calculated based on the gradient histogram, the histogram of the image was computed after it had undergone directional non-maximum suppression. As discussed in Section 2, an 8-step non-uniform quantizer was employed to obtain the discrete histogram for each processed block. The block-based hysteresis thresholds (high threshold ThH and low threshold ThL) were computed.

E. Thresholding with hysteresis:

Since, the output of the non maximum suppression unit contains some spurious edges, the method of thresholding with hysteresis was used. Two thresholds, high threshold ThH and low threshold ThL, which were obtained from the threshold calculation unit, were employed. Let f(x, y) be the image obtained from the non maximum suppression stage, f1(x, y) be the strong edge image and f2(x, y) be the weak edge image. Fig. 4 illustrates the pipelined design of this stage.

IV. Simulation Results And Analysis

Implementation was done for Non-Maximum Suppression Unit and Thresholding Unit that had resulted in Canny edge

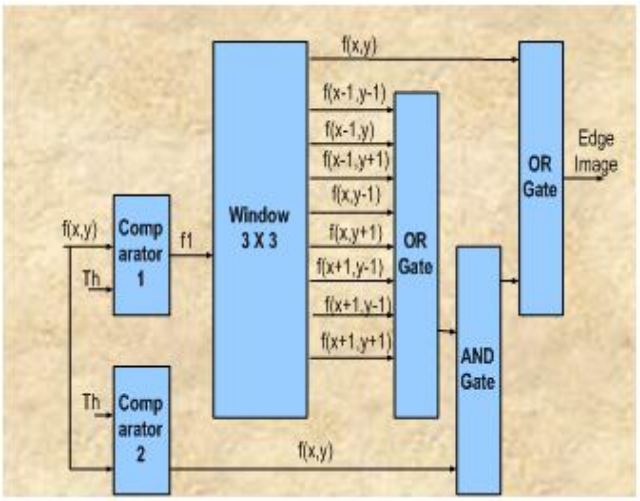

Figure 4. Pipelined architecture of the Thresholding Unit

detection system. The implementation was achieved with the device FPGA Virtex 5 XC5VTX240T and Package FF1759. Design utilization summary obtained for Non-Maximum Suppression Unit is given in Fig.5(a) and RTL top module schematic and simulation results of Non- Maximum Suppression Unit is shown in Fig. 5(b) and 5(c) respectively.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	11443	149760	7%
Numbers of Slice LUTs	26223	149760	17%
Number of fully used LUT-FF pairs	3440	34226	10%
Number of bonded IOBs	23	680	3%
Number of BUFG/BUFGCTRLs	3	32	9%

Figure 5 (a). Design Summary of Non-Maximum Suppression Unit

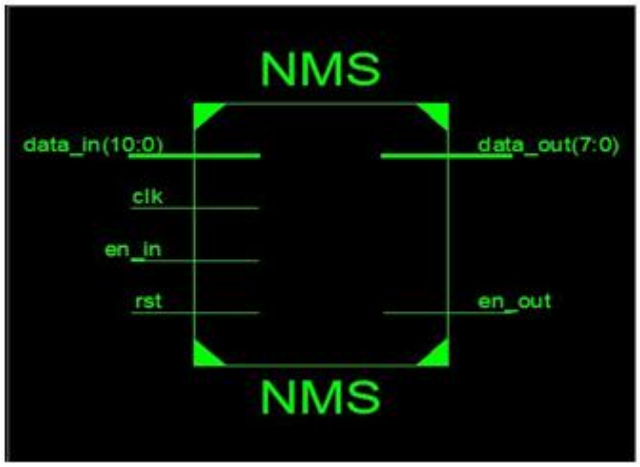

Figure 5 (b). RTL Top module Schematic of Non-Maximum Suppression Unit

Design utilization summary obtained for Thresholding Unit is given in Fig. 6(a) and RTL top module schematic and simulation results of Thresholding Unit is shown in Fig. 6 (b) and Fig. 6 (c) respectively.

DOI: 03.LSCS.2013.2.26

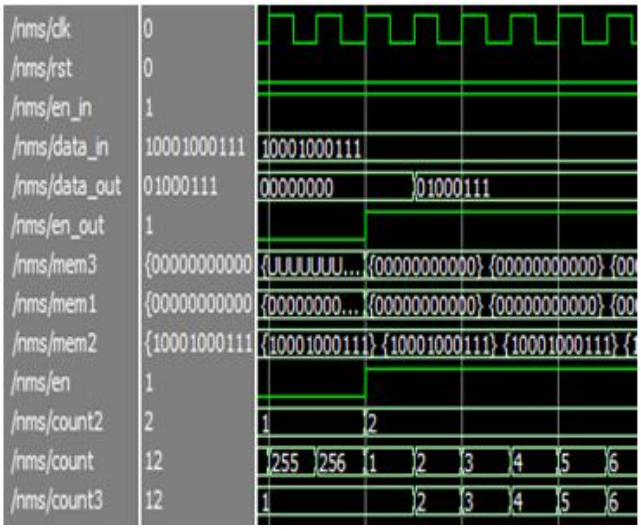

Figure 5 (c). Simulation Results of Non-Maximum Suppression Unit

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	193	149760	0%
Numbers of Slice LUTs	271	149760	0%
Number of fully used LUT-FF pairs	163	301	54%
Number of bonded IOBs	30	680	4%
Number of BUFG/BUFGCTRLs	2	32	6%

Figure 6 (a). Design Summary of Thresholding Unit

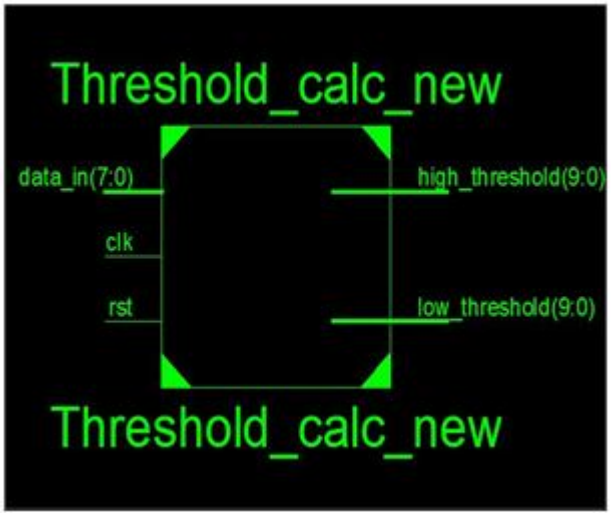

Figure 6 (b). RTL Top module Schematic of Thresholding Unit

Summary of Non-Maximum Suppression Unit and Thresholding Unit as shown in Table I below.

IV. Conclusion

A novel non-uniform quantized histogram calculation method was proposed in the present work, in order to reduce the computational cost of the hysteresis threshold selection for the edge detection. An architecture for Non-maximal Suppression used in Canny edge detection algorithm was also designed that gave significant result with respect to

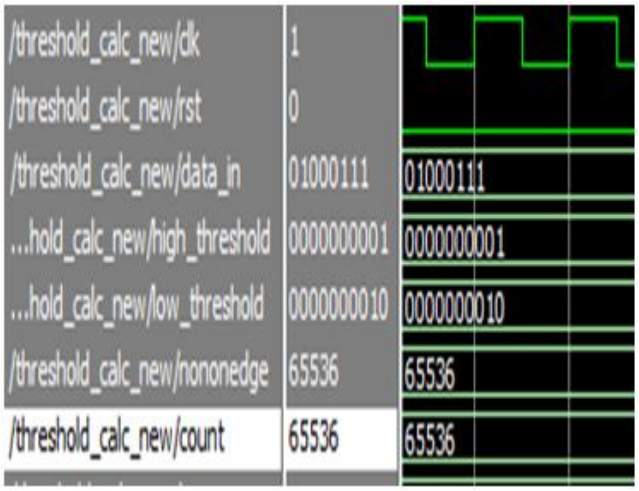

Figure 6 (c). Simulation Results of Thresholding Unit

Table: I. Non-Maximum Suppression Unit and Thresholding Unit

	Non-Maximum Suppression Unit			Thresholding Unit		
Slice Logic Utilization	Used	Available	Utilization	Used	Available	Utilization
Number of Slice Registers	11443	149760	7%	193	149760	0%
Number of Slice LUTs:	26223	149760	17%	271	149760	0%
Number of fully used LUT-FF pairs	3440	34226	10%	163	34226	54%
Number of bonded IOBs	23	680	3%	30	680	4%
umber of BUFG/BUF GCTRLs	3	32	9%	2	32	6%
Latency	0.001051ms			0.001051ms		
Total time period	3.9332ms			3.9332ms		

reduction in memory, decrease in latency and increase in throughput with no loss in edge detection. As a result, it was found that the computational cost of the proposed algorithm is very low compared to the original Canny edge detection algorithm. The algorithm was mapped onto a Xilinx Virtex-5 FPGA platform and tested using ModelSim. It was observed that this was capable of supporting fast real-time edge detection for images and videos with various spatial and temporal resolutions including full-HD content.

References

[1] L. Torres, M. Robert, E. Bourennane, and M. Paindavoine, "Implementation of a recursive real time edge detector using retiming techniques," *VLSI*, pp. 811 –816, Aug. 1995.

[2] Qian Xu, Chaitali Chakrabarti and Lina J. Karam,"A Distributed Canny Edge Detector and Its Implementation On FPGA", Arizona State University, Tempe, AZ 2011.

DOI: 03.LSCS.2013.2.26

[3] S. Varadarajan, C. Chakrabarti, L. J. Karam, and J. M.Bauza, "A distributed psycho-visually motivated Canny edge detector," IEEE ICASSP, pp. 822 –825, Mar. 2010.

[4] Christos Gentsos, Calliope-Louisa Sotiropoulou and Spiridon Nikolaidi,"Real- Time Canny Edge Detection Parallel Implementation for FPGAs",499-502,ICECS,12-15 Dec. 2010.

[5] W. He and K. Yuan, "An improved Canny edge detector and its realization on FPGA," WCICA, pp. 6561 –6564, Jun. 2008.

[6] D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," ITCC, vol. 2, pp. 843 – 847, Apr.2004.

[7] Shengxiao Niu, Jingjing Yang, Sheng Wang, Gengsheng Chen,"Improvement and Parallel Implementation of Canny Edge Detection Algorithm Based on GPU". ASIC (ASICON), 2011.

[8] J. Canny, "A computational approach to edge detection,"IEEE Trans. PAMI, vol. 8, no. 6, pp. 679 –698, Nov. 1986.

[9] Anand Gupta, Ravi Kumar Dalal,Rahul Gupta,Pulkit Wadhwa "DOW-Canny: An Improvised Version Of Canny Edge Detector". (ISPACS) December 7-9, 2011.

DOI: 03.LSCS.2013.2.26